第3章 FPGA設計を体験する

――回路の設計，IPコアの活用，実機による検証

剣持裕治，森あきら

ここでは，本誌2005年1月号に付属したFPGA基板を用いて簡単な回路を設計し，FPGAを使って実際に動作させながら検証を行います．また，マイクロプロセッサのIPコアを使用した設計とデバッグも体験します．開発したFPGAを動作させるために必須の基板設計についても触れます．（筆者）

本稿では，FPGAの回路設計から検証，基板設計の一連の工程を体験します．開発ツールとして，オーストラリアAltium社の「DXP 2004」製品群に含まれるFPGA開発のための「Nexar 2004」とプリント基板開発のための「Protel 2004」を使用します．これらのツールについては，無償で使用できる評価版（ライセンス登録は必要）が用意されています（pp.40-41のコラム「DXP 2004評価版のインストールとライセンス認証」を参照）．

また，あらかじめ用意されているIP（Intellectual property）コア（NEXUSコンポーネント）による設計・デバッグ手法についても説明します．なお，本記事で紹介した設計データやプロジェクト・ファイルは，本誌のホームページ（http://www.cqpub.co.jp/dwm/）からダウンロードできます．

1．FPGAを動作させてみよう

剣持裕治

まず，簡単な回路を設計し，FPGAに実装して動作させます．また，FPGAボードを使って動作の検証を行います（pp.42-43のコラム「FPGAボードの準備」を参照）．

FPGAの設計

ここではクロックでカウントアップするカウンタ回路を設計します．カウンタの出力で8個のLEDを点灯させます．

DXPを起動すると**図1**の画面になります．

● プロジェクトの作成

設計データはプロジェクトという単位で管理します．まず，新しいプロジェクトを作成します．

DXPのメニューから「ファイル（F）」→「新規（N）」→「Project」→「FPGA Project」を選択すると，**図2**のウィンドウが開きます．プロジェクト名の「FPGA_Project1.PrjFpg」



図1　DXPを起動してファイルを読み込んだ状態
画面左側のパネルではプロジェクトが表示されている．右側端の「ライブラリ」タブをクリックすると，ライブラリ・パネルが開く．

コラム　DXP 2004評価版のインストールとライセンス認証

● インストールについて

ここで使用している「Nexar 2004」と「Protel 2004」は，Altium社の「DXP 2004」に含まれるツールです．DXP 2004の評価版は，本誌2005年4月号の付属CD-ROMに収録されています．お持ちでない方は，アルティウムジャパンのホームページ(http://www.altium.co.jp/)からダウンロードすることができます．

DXP 2004の動作環境を**表A-1**に示します．インストールには，Windows 2000/XPのAdministrator権限が必要です．会社で支給されているパソコンなどでAdministrator権限がない方は，システム管理者に問い合わせてください．

DXP 2004のファイルが収録されたフォルダにあるsetup.exeを実行します(Altium社のCD-ROMではオートランも可能)．インストール・フォルダはデフォルトではC:\Program Files\Altium2004ですが，C:\Altium2004にしたほうが使いやすくなります．

インストールが終了したら，Windowsのデスクトップに作成されたDXP 2004のアイコンをダブルクリックして起動し，基本的な設定を行います．まず，ダイアログとメニューの日本語を有効にします．メニューから「DXP」→「Preferences」で開くダイアログで，**図A-1**のようにLocalization部の3ヵ所をチェックし，［Apply］ボタンを押します．［OK］ボタンでダイアログを閉じた後，DXPを再度起動すれば，画面が日本語化されます．

表A-1　DXP 2004の動作環境

OS	Windows 2000 ProfessionalまたはWindows XP
CPU	2GHzのPentium 4または同等品以上(3GHz以上を推奨)
メイン・メモリ	512Mバイト(1Gバイト以上を推奨)
ハード・ディスク	2Gバイト以上
グラフィック	1280×1024ピクセル以上 (640×480ピクセル以上のセカンド・モニタの利用を推奨)
そのほか	CD-ROMドライブ(インストール用) インターネット接続環境と電子メール・ソフト，Webブラウザ(ライセンス認証用) パラレル・ポート(USB接続やPCカードのポートは不可) ISE WebPACK 6.3i(Xilinx社のFPGAを使用する場合)

●ライセンス認証

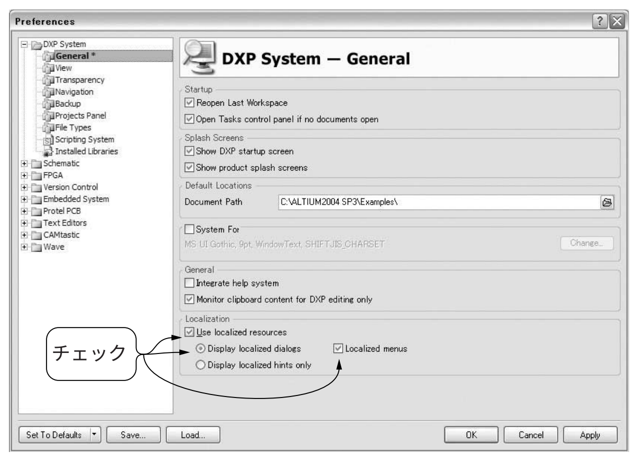
DXP 2004を利用するためには，ライセンス認証が必要です．

電子メールでDesignWave@altium.co.jpに必要事項(会社名，部署名，役職，氏名，郵便番号，住所，電話番号，FAX番号，E-mailアドレスなど)を知らせると，Customer NumberとActivation Codeが送付されてきます．ライセンス認証にはこの二つのコードが必要です．

ライセンス認証手続きは，Web経由を推奨しています．インターネットに接続した状態でDXP 2004を起動すると，**図A-2**の画面が表示されます．もし画面が違っていたら，「DXP」→「ライセンス(L)」で画面が開きます．「web利用のライセンス認証」の部分をマウスでクリックすると，ライセンス認証に関する注意事項が表示されます．承諾する場合はチェックを付けて，［OK］ボタンを押します．

図A-3のAltium社のライセンス認証ページであらかじめ入手しておいたCustomer NumberとActivation Codeを入力し，［Yes］ボタンを押します．

図A-4のユーザ登録のページで，必要事項を記入します．*マークの項目の記入は必須です．すべて記入したら，［Next］ボタンを押し



図A-1　DXP 2004の設定画面
Localization部の3ヵ所にチェック．

は任意の名まえに変更することができます．

● 回路図の作成

LSI設計ではHDLが用いられますが，最上位階層については回路図が使われることがよくあります．最上位階層の回路は機能ブロック図に近くなり，回路の働きが把握しやすくなります．DXP 2004では，最上位階層は回路図で表現することになっています(詳しくはp.45のコラム「VHDLやVerilog HDLを使う」を参照)．

メニューから「ファイル(F)」→「新規(N)」→「Schematic」



図2　新規プロジェクトの作成
設計データはプロジェクトという単位で管理される．

ます．問題がなければ，終了ページが表示されるので，［Finish］ボタンを押します．

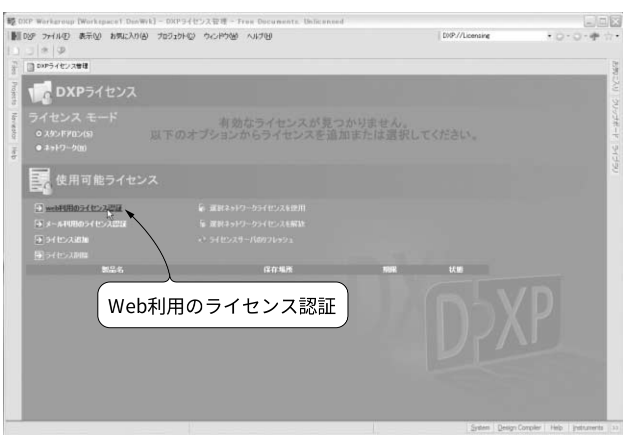
ここで記入したE-Mailアドレスあてにライセンス・ファイルが添付ファイルとして送られてきます．必要に応じて，メール・ソフトウェアなどの設定を行ってください．ライセンス・ファイルの拡張子は.alfで，ファイルの内容はASCIIファイルです．このファイルをDXP 2004のインストール・フォルダに保存してください．

ライセンス画面（**図A-2**）に戻り，「ライセンス追加」をクリックすると，ライセンス・ファイルを保存したフォルダを問い合わせてきます．保存したファイルを選択して［開く（O）］ボタンを押せば，ライセンス認証が完了します．

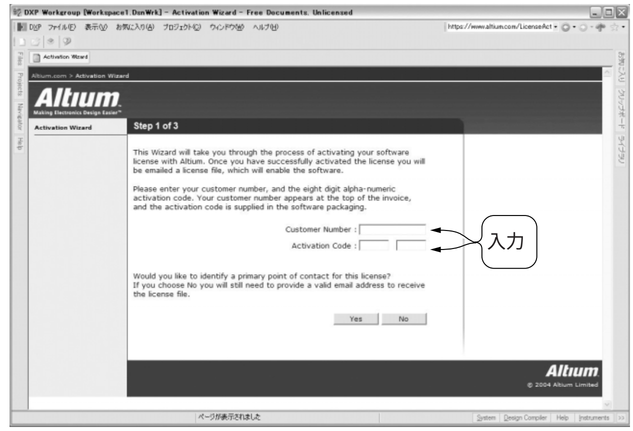
評価版のライセンスは，ライセンス認証を行ってから30日の期限があることに注意してください．期限が過ぎるとファイルの保存などができなくなります．

なお，ここでは本記事執筆中にベータ・リリースされたDXP 2004サービスパック3を使用しています．

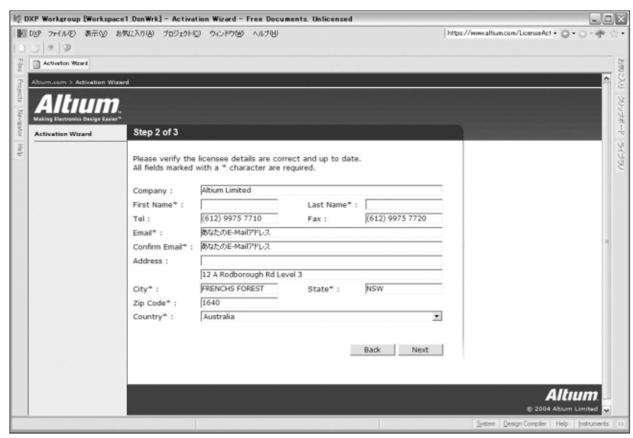
越智 誠　アルティウムジャパン（株）



図A-2　ライセンス画面
「web利用のライセンス認証」でライセンス認証手続きを行う．



図A-3　ライセンス認証ページ
あらかじめ入手しておいたCustomer NumberとActivation Codeを入力する．



図A-4　ユーザ登録のページ
*マークの項目は必須．ここで記入したE-Mailアドレスあてにライセンス・ファイルが添付ファイルとして送られてくる．

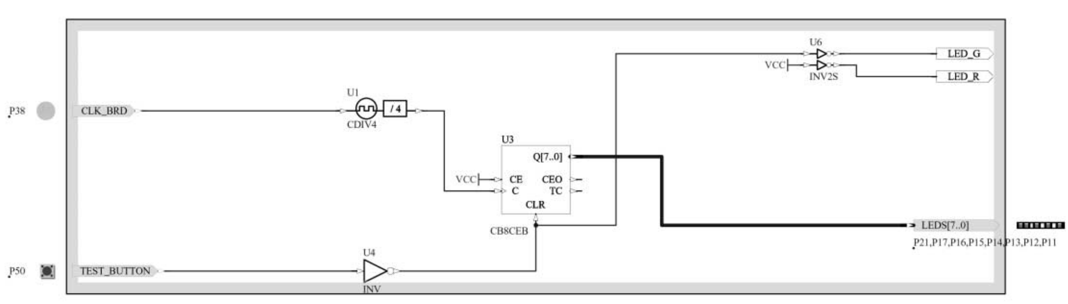


図3　カウンタ回路
CB8CEBは8ビット・カウンタ．クロックによりカウント・アップする．スイッチでカウンタがリセットされる．

コラム　FPGAボードの準備

FPGAは単なるLSIであり，単体では動作しません．本文で説明しているような実機を用いた検証を行うためには，電源回路，クロック，NEXUSインターフェース用のコネクタかパソコンのパラレル・ポートへ接続するためのインターフェースのあるボードが必要です．

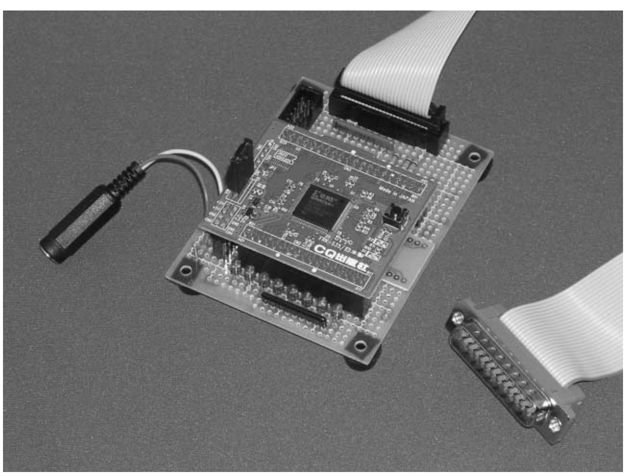
● 本誌2005年1月号の付属FPGA基板を利用する

本誌2005年1月号の付属FPGA基板には3.3V供給用の「LM317T」とタクト・スイッチ用のパターンが用意されているので，これらを実装します．ベース・ボードについては，必要に応じてLEDやスイッチなどをFPGAのI/Oピンと接続します．今回は**図B-1**のような回路にしました．製作したボードを**写真B-1**に示します．

クロック発振器は，手持ちの部品の中から12MHzのものを使いました．今回の設計では，周波数は重要ではないので，入手しやすい値のものを使いました．

74LCX244はパソコンとのインターフェース回路です． Altium社の「Universal JTAG Interface」を持っている場合は，この回路を省略できます．一般的なJTAGインターフェースはJTAGが1系統のため，今回の事例には不向きです．

パソコンのパラレル・ポートは25ピンのDサブ・コネクタです．FPGAボード側は，コネクタの指定はありません．ユニバーサル基板で作成する場合は，Dサブより**写真B-1**のようなコネクタのほうが製作しやすいでしょう．



写真B-1 製作したボード

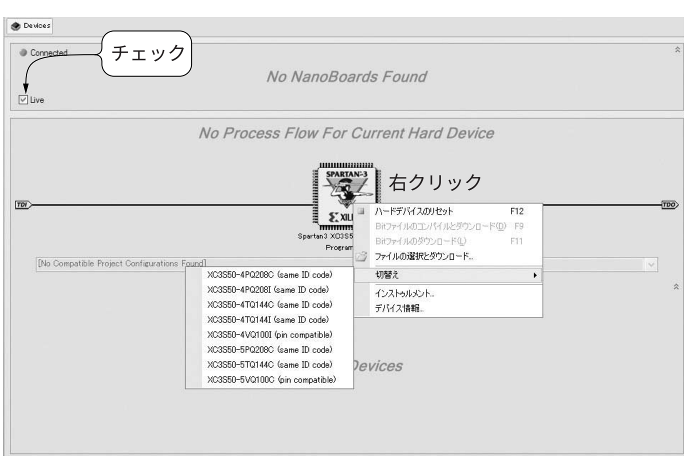
● FPGAボードとパソコンの接続の確認

FPGAボードとパソコンを接続します．DXP 2004のメニューから「表示(V)」→「デバイスビュー(V)」を選択して，デバイス・ビュー画面を表示します．左上のLiveチェック・ボックスを有効にすると，**図B-2**のようにFPGAが認識されます．FPGAの型名が異なっている場合は，デバイスのアイコンの上で右クリックし，正しい型名を選択します．これは，FPGAが返してくるIDだけでは厳密な型名まで区別できないためです．

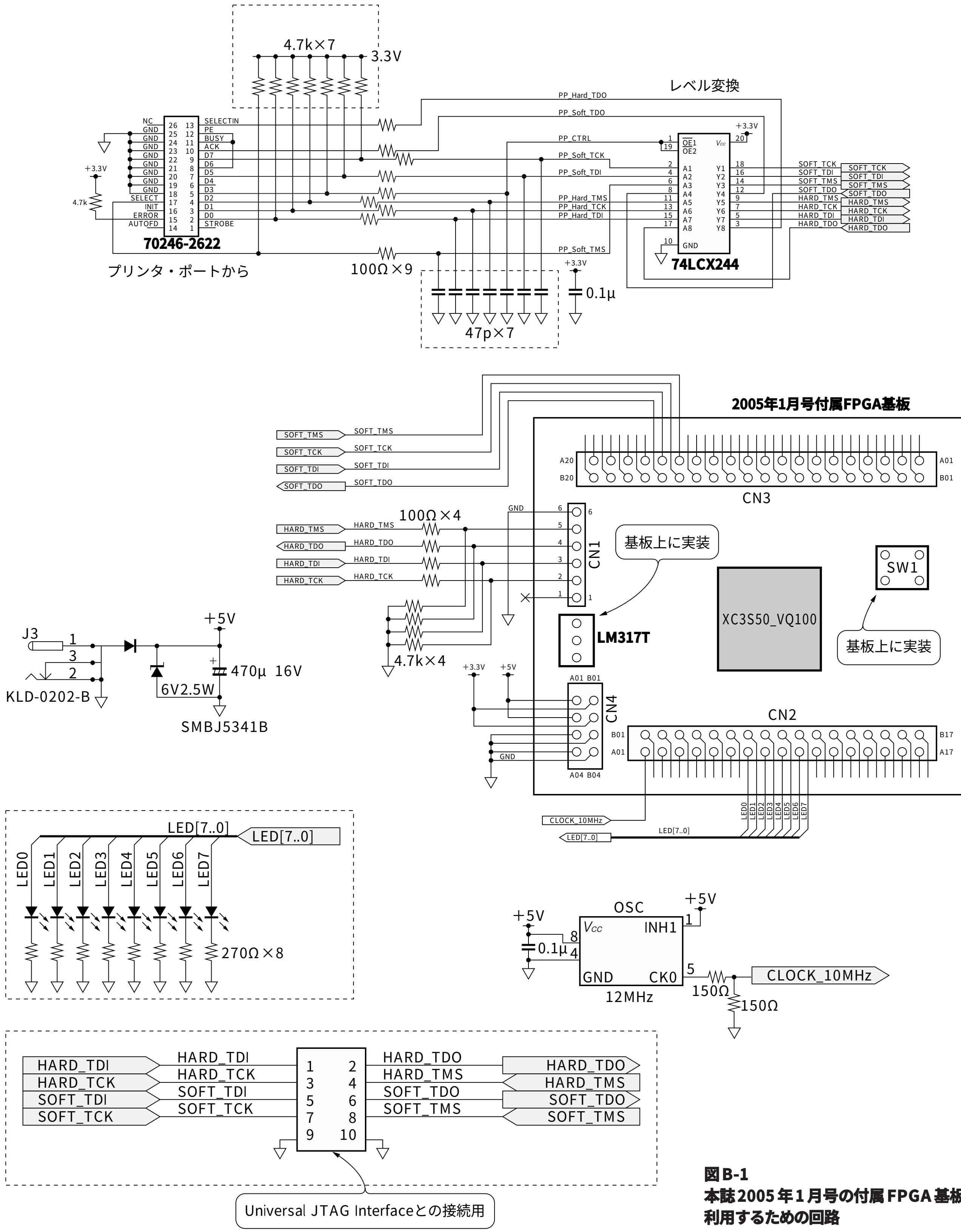
デバイスのアイコンをダブルクリックして開く画面の「Live更新」をチェックすると，バウンダリ・スキャンによってピンの状態を見ることができます(**図B-3**)．FPGAには回路が書き込まれていない状態ですが，38番ピンのクロックの入力と50番ピンのスイッチによる信号の変化を観察できます．



図B-3　バウンダリ・スキャンによるピンの状態表示
「Live更新」をチェックする．信号の変化を観察できる．



◀ **図B-2**
デバイス・ビュー画面
正常に接続されていれば，FPGAが認識される．FPGAの型名が異なっている場合は，デバイスのアイコンの上で右クリックし，正しい型名を選択する．



図B-1
本誌2005年1月号の付属FPGA基板を利用するための回路

リスト1　制約ファイル

```
;..........................................................................
Record=FileHeader | Id=DXP Constraints v1.0
;..........................................................................

;..........................................................................
Record=Constraint | TargetKind=PCB  | TargetId=DesignWave JAN.2005
Record=Constraint | TargetKind=Part | TargetId=XC3S50-4VQ100C
;..........................................................................

;..........................................................................
; OnBoard LEDs
;..........................................................................
Record=Constraint | TargetKind=Port | TargetId=LED_G              | FPGA_PINNUM=P47
Record=Constraint | TargetKind=Port | TargetId=LED_R              | FPGA_PINNUM=P49

;..........................................................................
; OnBoard SWITCH
;..........................................................................
Record=Constraint | TargetKind=Port | TargetId=TEST_BUTTON        | FPGA_PINNUM=P50

;..........................................................................

; Option LEDs
;..........................................................................
Record=Constraint | TargetKind=Port | TargetId=LEDS[7..0]         | FPGA_PINNUM=P21,P17,P16,P15,P14,P13,P12,P11

;..........................................................................
; Clocks
;..........................................................................
Record=Constraint | TargetKind=Port | TargetId=CLK_BRD            | FPGA_PINNUM=P38
Record=Constraint | TargetKind=Port | TargetId=CLK_BRD            | FPGA_CLOCK=TRUE
Record=Constraint | TargetKind=Port | TargetId=CLK_BRD            | FPGA_CLOCK_FREQUENCY=12 Mhz
;..........................................................................
```

を選択し，新規の回路図を作成します．今回は，**図3**に示した回路図を入力します．必要な部品（コンポーネント）は，画面の右側に表示されている「ライブラリ・パネル」タブから選択して配置します（**図4**）．ここで使用するライブラリは，FPGA Generic.IntLibとFPGA NanoBoard Port-Plugin.IntLibです．

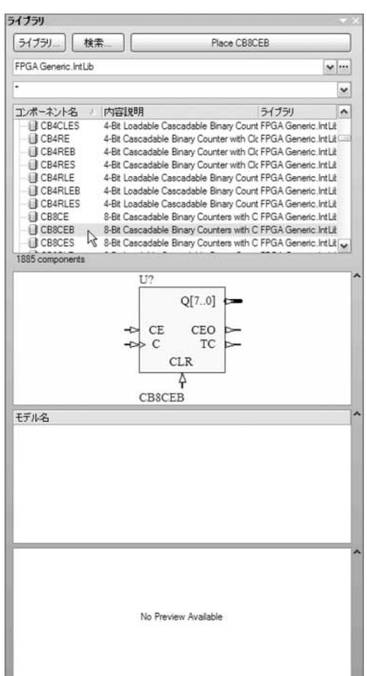


図4
ライブラリ・パネル
回路図で使用する部品（コンポーネント）をこの画面で選択する．

外部との入出力はポートというコンポーネントを使います．CB8CEBは8ビット・カウンタのコンポーネント，INVはNOTゲートのコンポーネント，CDIVは4分周するコンポーネントです．コンポーネント間は配線を使って接続していきます．バス（複数の配線のまとまり）やワイヤ（1本の配線）を使います．

図3の回路は，外部から供給されたクロック（CLK_BRD）を4分周し，8ビットの出力をLEDに出力（LEDS［7..0］）する回路になります．

● 制約ファイルの作成

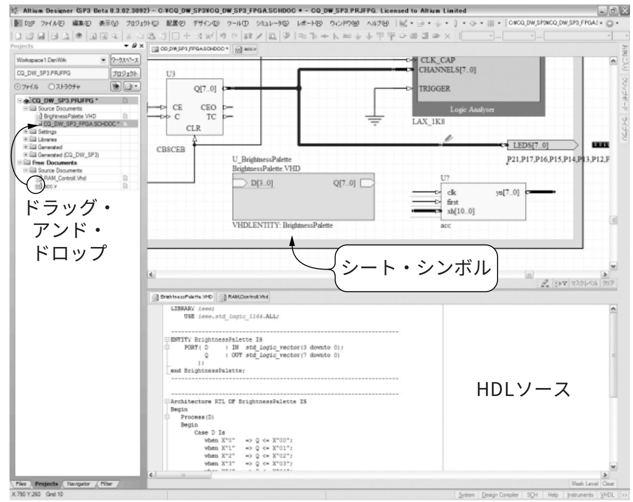
FPGAのI/Oピンのピン番号とFPGAに実装する回路の信号名を対応づけるため，制約（コンストレイント）ファイルを作成します．

メニューから「ファイル（F）」→「新規（N）」→「Other」→「Constraint File」を選択します．ここでは**リスト1**に示す制約ファイルを作成します．制約ファイルへのピン番号の記述は，「デザイン（D）」→「コンストレイントの追加/修正（A）」→「ポート」を選択すると表示されるAdd/Modify

コラム　VHDLやVerilog HDLを使う

DXP 2004によるFPGA設計では，最上位階層は回路図で表記する必要があります．一方，それ以下の階層ではハードウェア記述言語を使用できます．VHDLについてはVHDL87とVHDL93，Verilog HDLではVerilog 1995とVerilog 2001の規格をサポートしています．

プロジェクト内にそれらの言語をドラッグして貼り付け，プロジェクト・パネルで右クリックし，「新規ドキュメントをプロジェクトに追加(N)」→「VHDL Document」(または「Verilog Document」)を選択します．テキスト・エディタが起動するので，HDLコードを入力します．すでにHDLソース・ファイルがある場合は，「ファイル(F)」→「開く(O)...」を選択してソース・ファイルを指定すると，テキスト・エディタが起動し，ソースが表示されます．しかし，このままではプロジェクトに属さない孤立したドキュメントの状態です．これをプロジェクトに加えるには，プロジェクト・パネル内に表示されているドキュメントの名まえの部分にあるアイコンをドラッグし，プロジェクトの範囲内にドロップします．そして，「デザイン(D)」→「シートからシートシンボルを作成(Y)」を選択すると，シート・シンボル(下位階層を持つブロックの記号)が現れます(**図C-1**)．また，「デザイン(D)」→「ファイルから回路図パート作成(S)」を選択すると，回路図ライブラリのシンボルとしてHDLソースを抽象化できます．こうすることで，HDLで記述されたブロックをほかの回路図シンボルと同じように取り扱えるようになります．



図C-1　HDLで記述されたブロックを回路図で使用
VHDLソース(BrightnessPalette.VHD)をシート・シンボル化して回路図に表示したようす．HDLソースをプロジェクトに加えるためには，Free DocumentsにあるHDLソース・ファイルをプロジェクトの領域にドラッグ・アンド・ドロップする．

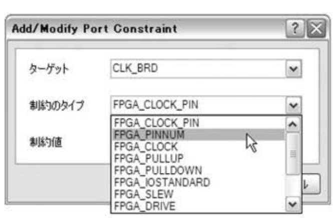


図5
制約ファイルの作成
制約ファイルへのピン番号制約はAdd/Modify Port Constraintダイアログで行える．図はポート名CLK_BRDに制約タイプFPGA_PINNUM(ピン番号)を追加しているようすである．制約タイプはプルダウン・メニューに用意されている．

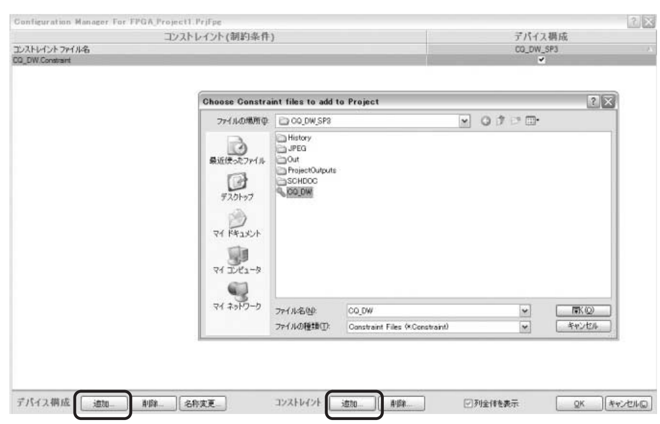


図6　デバイス構成の設定
「デバイス構成」の[追加...]ボタンを押し，適当な名まえを入力する．次に「コンストレイント」の[追加...]ボタンを押して，制約ファイルを登録する．

Port Constraintダイアログから行えます(**図5**)．あらかじめ作成済みの制約ファイル(.ucfなど)を読み込むには，「デザイン(D)」→「Pinファイルのインポート…」を選択します．

リスト1では，例えば「P38」というピンにはクロック信号(CLK_BRD)を割り当てています．

● デバイス構成の設定

制約ファイルと回路図を関連づけます．回路図を表示している状態で「プロジェクト(C)」→「コンフィグレーション・マネージャ」を選択すると，**図6**の画面が開きます．下のほうにある「デバイス構成」の[追加...]ボタンを押し，適当な名まえを入力します．次に「コンストレイント」の[追加...]ボタンを押して，制約ファイルを登録します．

1枚のプリント基板に複数のFPGAが搭載されることがあります．このようなときは，FPGAごとに用意した複数の制約ファイルを関連づけ，一つのプロジェクトで管理し

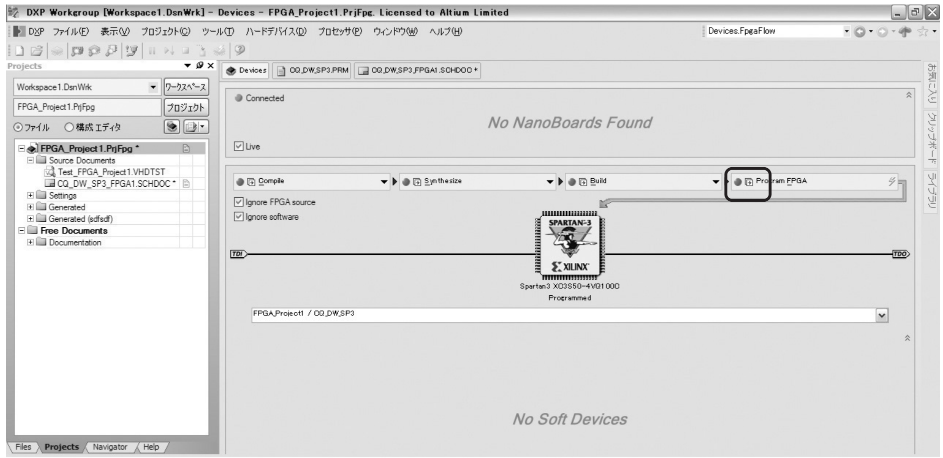


図7
FPGAへの回路の実装
「Program_FPGA」のアイコンをクリックすると，回路図などからFPGAの回路データが合成され，ボード上のFPGAへの回路データの書き込みまでが自動的に実行される．

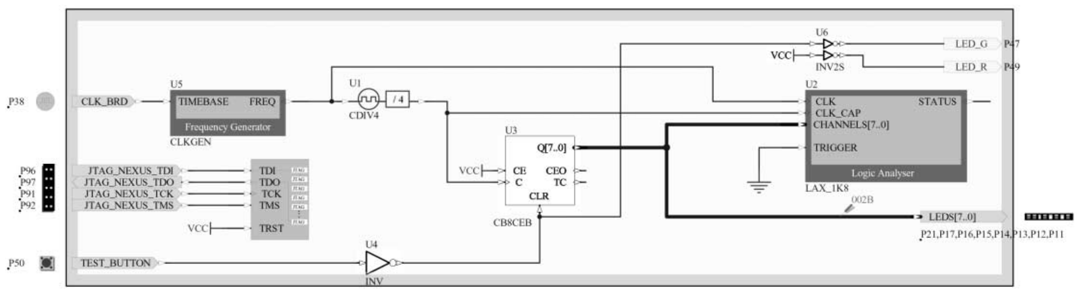


図8
計測器と信号発生器を埋め込んだカウンタ回路
CLKGENが信号発生器，LAX_1K8が計測器である．これらの機能を使うためには，4本の信号を使うJTAGインターフェースを使用する．

ます．

● **FPGAへの回路の実装**

DXPのプロジェクト画面に戻ります．FPGAボードをパソコンに接続し，FPGAが正しく認識されていると**図7**のような画面になります．ここで「Program_FPGA」のアイコンをクリックすると，回路図などからFPGAの回路データが合成され，ボード上のFPGAへの回路データの書き込みまでが自動的に実行されます．

もしエラーがあると，処理が中断します．表示されたメッセージを参考に，エラーを修正します．

エラーの表示がなく終了すれば，FPGAが動作を始めます．

実機を使って動作を検証する

FPGAボードに8個のLEDがあれば，すべて点灯しているように見えると思います．実際にはカウンタとして動作しているのですが，高速に点灯/消灯が繰り返されているため，残像効果ですべて点灯しているように見えるのです．

これでは正常に動作しているのかどうかわかりません．

LSI設計では，設計した回路が正しく動作するかどうかをシミュレーションなどにより検証します（右掲のコラム「シミュレーションによる検証」を参照）．シミュレーションは，とても時間がかかるという欠点があります．そこでFPGAを使った設計では，実機を使った検証をよく行います．ここでは，ロジック・アナライザなどの計測器を使って動作を確認します．

● **FPGAに計測器を埋め込む**

ここではFPGA独特の検証方法を説明します．回路の中に，ロジック・アナライザと信号発生器の機能を埋め込んでしまうものです．まさに回路構成を自由に変更できるFPGAの特徴を生かした検証手法と言えます．もちろん，ASIC設計でも試作においてFPGAを活用する場合には適用可能な手法です．組み込みプロセッサ・デバッグ用の標準インターフェースNexus 5001（IEEE-ISTO 5001-1999）[注1]とJTAGインターフェース（IEEE 1149.1）によってパソコンと通信します．

回路を**図8**のように変更します．FPGA Instruments.IntLibから新たに追加したコンポーネントは，CLKGENと

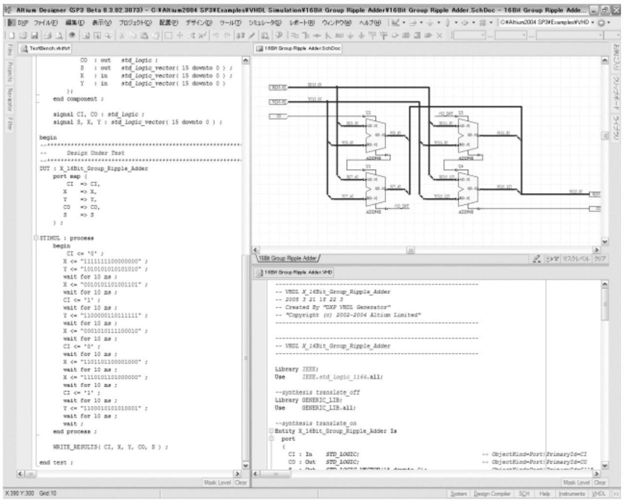
注1：詳細はhttp://www.nexus5001.org/を参照．

コラム　シミュレーションによる検証

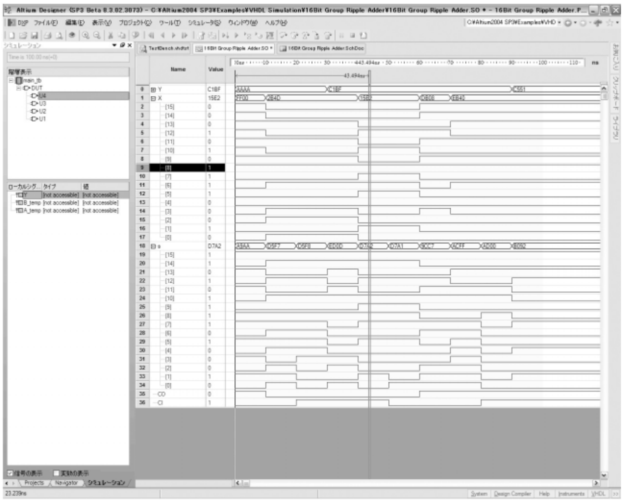
本文では，FPGAを実際に動作させて検証を行っていますが，ここではLSI設計において一般的な，シミュレーションによる検証について説明します．

まず，検証したい回路を用意します．検証に不必要な部分は「配置(P)」→「ディレクティブ(I)」→「コンパイルマスク」を選択して，コメント扱いにしておきます．次に，「シミュレータ(S)」→「VHDLのコンパイル(C)」を選択し，VHDLファイルを作成します．作成されたVHDLファイルは，プロジェクト・パネルのGenerated内に保存されています．VHDLファイルをダブルクリックすると，ファイルが開きます．ファイルを開いた状態で，「デザイン(D)」→「VHDLテストベンチ作成」を実行し，テストベンチを作成します(**図D-1**)．

「シミュレータ(S)」→「シミュレート(S)」を選んで必要な時間だけシミュレーションを実行すれば，波形エディタにシミュレーション波形が表示されます(**図D-2**)．Altium2004 \ Examples \ VHDL Simulationに各種のサンプルが格納されているので，いろいろと試してみてください．



図D-1　シミュレーションの設定
右上が回路図，右下が生成されたVHDL，左側はテストベンチ．



図D-2　シミュレーション結果
「シミュレータ(S)」→「シミュレート(S)」を選択して得られたシミュレーションの結果の例．

リスト2
制約ファイルへのピンの追加

```
; NEXUS JTAG Soft-Device Chain Connections
;..............................................................................
Record=Constraint  | TargetKind=Port | TargetId=JTAG_NEXUS_TCK       | FPGA_PINNUM=P91
Record=Constraint  | TargetKind=Port | TargetId=JTAG_NEXUS_TDI       | FPGA_PINNUM=P96
Record=Constraint  | TargetKind=Port | TargetId=JTAG_NEXUS_TDO       | FPGA_PINNUM=P97
Record=Constraint  | TargetKind=Port | TargetId=JTAG_NEXUS_TMS       | FPGA_PINNUM=P92
;..............................................................................
```

LAX_1K8の2種類の仮想測定器です．この仮想測定器を外部と接続するため，FPGA Generic.IntLibからNEXUS_JTAG_PORTを追加します．新しい信号を追加するため，制約ファイルに**リスト2**の記述を追加します．

「ツール(T)」→「番号割り付けの実行(U)」を選択し，コンポーネントをほかと区別するための名まえを割り当てます．

FPGAへの回路の実装を再び実行します．するとデバイスの絵の下側に，新しいチェーンが現れます(**図9**)．上のチェーンは，FPGAが持つ本来のJTAGチェーンです(ハードJTAGチェーンと呼ぶ)．下のチェーンは，新たに追加したJTAGチェーンです(ソフトJTAGチェーンと呼ぶ)．ソフトJTAGチェーンには，追加した仮想計測器が2個並んでいます．

ハードJTAGチェーンにあるFPGAの絵をダブルクリックすると，埋め込んだ計測器を操作できます(**図10**)．

まず，信号発生器を操作してみましょう．[Set Time Base]ボタンを押し，FPGAに供給するクロック周波数を入力します(**図11**)．この周波数が計測の基準となる周波数です．

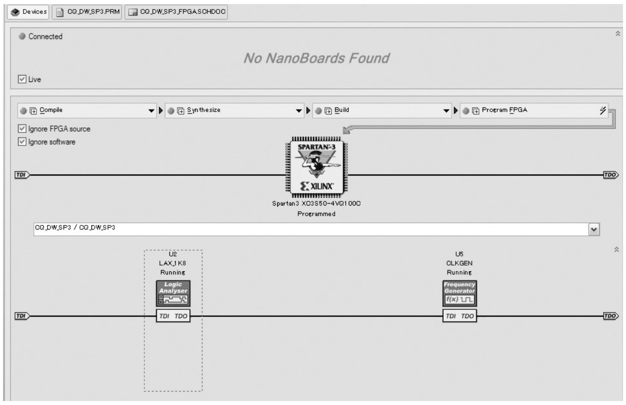


図9　仮想計測器の追加

上のチェーンは，FPGAが持つ本来のJTAGチェーン（ハードJTAGチェーンと呼ぶ）．下のチェーンは，新たに追加したJTAGチェーン（ソフトJTAGチェーンと呼ぶ）．ソフトJTAGチェーンには，追加した仮想計測器が2個並んでいる．

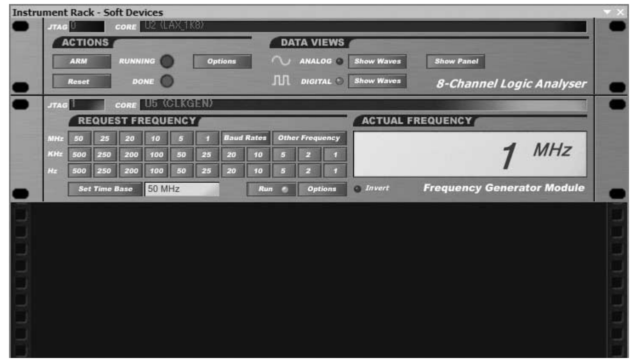


図10　FPGAに埋め込まれた計測器のユーザ・インターフェース

上がロジック・アナライザ，下が信号発生器である．



図11　クロック周波数の設定

クロック周波数を任意に変更することができる．

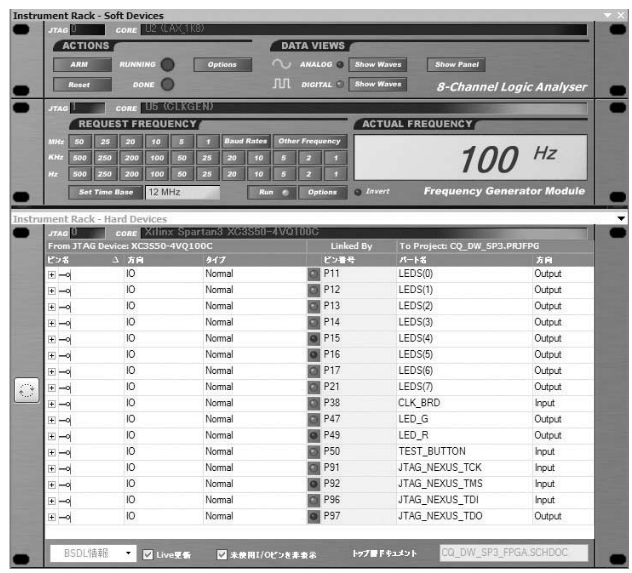


図12　FPGAのI/Oピンの状態を表示

FPGAの絵をダブルクリックすると，FPGAのI/Oピンの状態を画面で見ることができる．

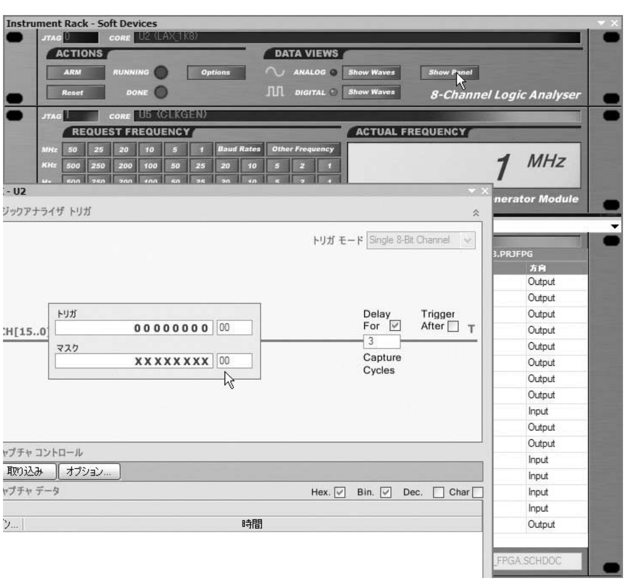


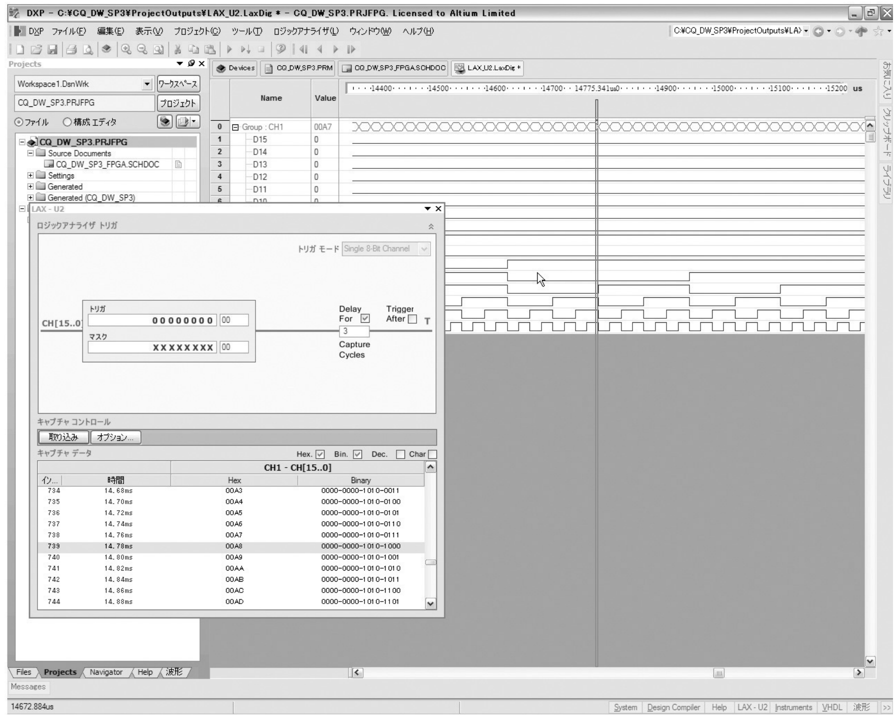
図13　トリガの設定

常時トリガの場合は，「マスク」を「00」にする．

周波数を100Hz程度まで落としてみると，LEDの点滅が目で見える速度になると思います．LEDを実装していなくても，FPGAの絵をダブルクリックすると，FPGAのI/Oピンの状態を画面で見ることができます（**図12**）．

次に，ロジック・アナライザで観察してみます．［Show Panel］ボタンをクリックし，計測開始の信号（トリガ）の設定を行います．今回は，**図13**のように「マスク」の欄に「00」を入力し，常時トリガ（つねに計測する）にしておきます．［取り込み］ボタンを押すと，波形の取り込みが始まります（**図14**）．波形を表示しきれていない場合は，［PageUp］，［PageDown］キーか［Ctrl］キーをクリックしながらマウスのホイール操作で見やすいサイズに拡大・縮小します．

図14
波形表示
ロジック・アナライザで観測された波形が表示される．信号発生器の周波数を1MHz程度まで上げてある．



▼**図15**
仮想計測器を埋め込んだ乗算回路
MULTU8Bが乗算回路．ディジタル入出力用の仮想計測器IOB_1X8（8ビット）とIOB_1X16（16ビット）を埋め込んでいる．

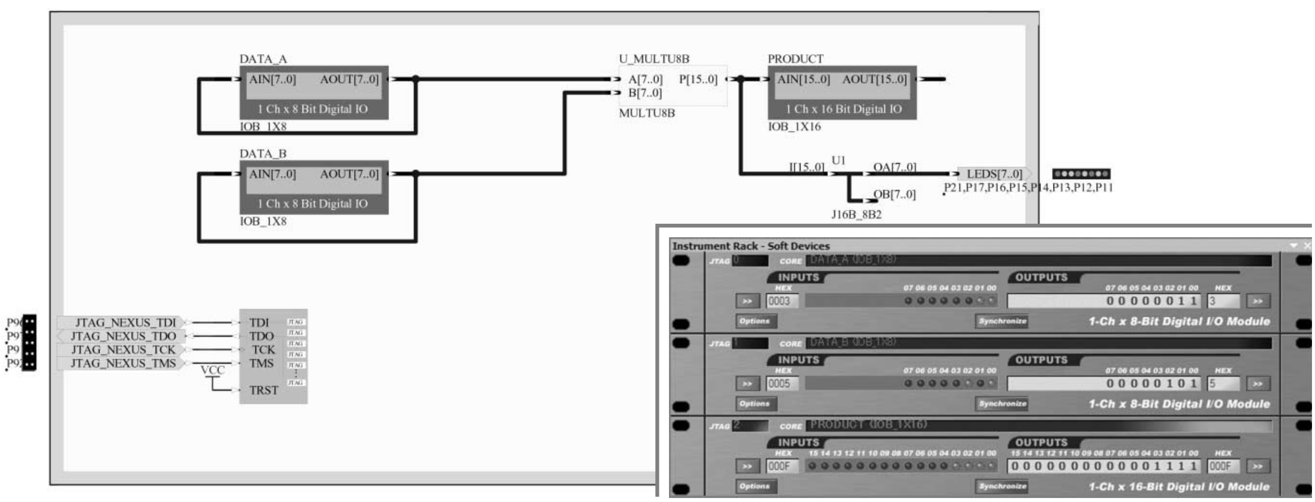


図16　仮想計測器を用いた乗算回路の検証
上の二つのパネルで入力信号を与えると，乗算結果が下のパネルに表示される．

乗算回路の設計と検証

乗算回路の設計と検証を行います．**図15**に回路を示します．A，Bそれぞれ8ビットの入力の計算結果をP［15..0］の16ビットから出力するものです．MULTU8Bが乗算回路です．

入出力にディジタル入出力用の仮想計測器を用いています．ここで使用するコンポーネントはFPGA Instruments.IntLibにあるIOB_1X8とIOB_1X16，FPGA Generic.IntLibにあるMULTU8Bです．ソフトJTAGチェーンを形成するNEXUS_JTAG_PORTも配置します．8個のLEDには，16ビット出力のうち下位8ビットだけを接続しておきます．制約ファイルとしては，**リスト1**に**リスト2**を追加したものをそのまま使用します．

FPGAに回路を実装し，仮想計測器を表示します．**図16**の画面で，上側にあるパネルから2進数で値を入力すれば，いちばん下のパネルに結果が表示されます．

2. ソフト・マクロのCPUコアを活用した設計の体験

森 あきら

CPUコアを搭載するLSIが増えています．FPGAであっても，集積度の向上に伴い，ソフト・マクロのCPUコアを使うことが珍しくなくなっています．

FPGAにソフト・マクロのCPUコアを搭載することそのものが目的となることはまずありません．複雑な処理のためにソフトウェアで設計したほうが簡単な場合や，仕様策定後のさまざまな要求にこたえる柔軟性が必要になるために，ソフト・マクロのCPUコアが使われるのです．

● 仕様

設計するLSIの仕様は，以下のとおりとします．

1）8個のLEDのうち，1番目を点灯させる．一定時間が経過したら消灯して，次のLEDを点灯する．これを繰り返して，8番目が終了したら1番目に戻る．

2）スイッチが押されているときには，LEDの点灯順序を逆にする．

3）LEDの点灯時間は，仮想計測器によって設定する．

このような機能は，ハードウェアの最初の動作確認時に要求されるものです．仕様の1）の機能により，CPUコアが機能していることと，出力ポート，LEDの点灯回路の動作を確認できます．仕様の2）の機能により，入力ポートとスイッチの動作を確認できます．仕様の3）の機能により，パソコンなどの外部機器とのインターフェースを確認できます．

● 回路の設計

使用するFPGAの規模は5万ゲート相当です．そこで，今回は，米国Microchip Technology社の「PIC16C5xシリーズ」と命令互換のCPUコアを使用します．使用ゲート数が少なく機能も限定されているため，今回のような用途に向いています．使用するIPコアは，DXP 2004に含まれます（p.52のコラム「CPUコアと実際のマイコンの違い」を参照）．

このCPUコアと仮想測定器のDigitalIOを用いた回路を**図17**に示します．プロジェクト名を「CQ_DW.PrjFpg」として回路図を作成してください．

「TSK165A_D」がCPUコアです．TSK165A_Dの末尾のDは，デバッグ・モジュールを意味します．デバッグ・モジュール付きのIPコアでデバッグを行い，最終的に製品に組み込む工程でデバッグ・モジュールのないIPコアに置き換えることもできます．このデバッグ・モジュールは，ソフトJTAGチェーンでパソコンと接続されます．

RAMS_12x512は命令コードと固定データを格納する12ビット×512バイトのシングルポート・メモリ，FPGA_STARTUP8はCPUコアにリセット信号を与えるプログラマブル遅延回路です．

FPGA基板上の2個のLEDは，スイッチの状態によって（CPUコアの動作と無関係に）点灯/消灯させます．こうすることで，不ぐあいが出たときに，FPGAの動作そのものが問題なのか，それともCPUコアの動作がおかしいのかという問題の切り分けに使うことができます．ただし，スイッチのON/OFF情報だけは，CPUコアの入力ポートRBI［7］（配線名はPB7）に接続しておきます．

CPUコアの入力ポートRCI［7..0］には仮想測定器の一つであるDigitalIOを接続します．これでパソコンの操作に

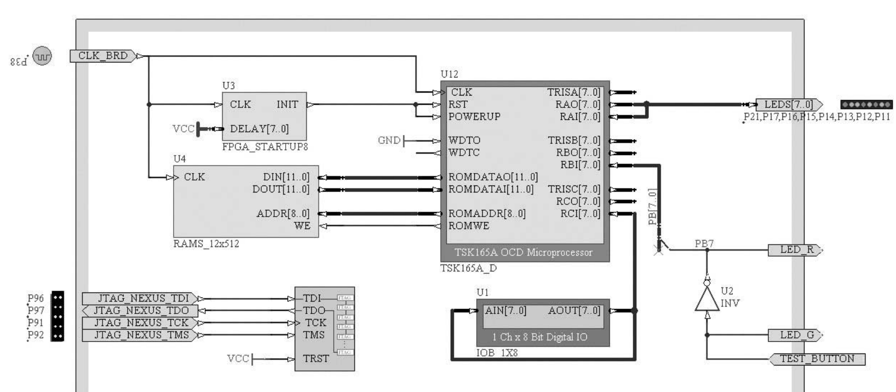


図17
CPUコアを用いた回路
「TSK165A_D」は，「PIC16C5xシリーズ」と命令互換のCPUコアである．

よって，CPUコアに8ビットの任意の定数を読み込ませることができるようになります.

● ソフトウェアの開発

CPUコアで実行するソフトウェアを**リスト3**に示します.

今回使用するCPUコアの命令メモリは512バイトです. データ・メモリは16＋9バイトのみです. 今回のソフトウェアで使用するデータ・メモリは，ソフト・タイマに2バイト，現在点灯しているLEDの位置を決めるメモリに1バイト，作業領域では1バイトです(**リスト3**の5～9行目).

メイン・プログラムでは，最初に使用するデータ・メモリの初期化とポートの初期化を行います. データ・メモリの初期化では，LEDposを0にします(**リスト3**の79行目). ポートの初期化では，出力ポートに初期データを出力します(**リスト3**の80行目). ほかのデータ・メモリについては初期化を行う必要がありません. 実際の処理のフローチャートを**図18**に示します. ソフトウェア・タイマと，LEDの点灯パターンを作成するサブルーチンがあります.

リスト3の54行目にあるラベルLongDelayからがソフトウェア・タイマです. 仮想測定器のDIGIAL I/Oからデータを入力して，タイマの時間を変化させています.

リスト3の21行目にあるラベルGetTableからがLEDの点灯パターンを作成するルーチンです. PICマイコンには，テーブルの索引命令がないので，Wレジスタに任意の値を戻す命令を繰り返し用意しておき，戻したいデータを持っている命令語にジャンプする方法をとっています. 基板上のボタンを押したとき，LEDが逆に動いているように見せるために，二つのテーブルを作成しておきます.

リスト3　サンプル・プログラム

```
001 ;.................................................
002 ; サンプル・プログラム
003 ;.................................................
004
005    .section Data, Data    ; RAM領域の定義
006 LEDPos        .dsb    1
                        ; 現在点灯しているLEDの位置を表している
007 CountInner    .dsb    1    ; ソフト・タイマ用のカウンタ下位
008 CountOuter    .dsb    1    ; ソフト・タイマ用のカウンタ上位
009 Wk            .dsb    1    ; 作業用領域
010
011 ;.................................................
012
013
014     .section Text, Code, inpage   ; 命令メモリ領域の定義
015     .global __start        ; 開始アドレスをグローバル定義
016 __start:
017     GOTO    MainProc
018
019 ;.................................................
020 ; W レジスタの下位4ビットより今回出力するLEDドライブ・データ
                                           を返すサブルーチン
021 GetTable:
022 ;.................................................
023     ANDLW    7
                 ; ビット位置指定に必要な下位3ビットを取り出す
024     MOVWF    Wk                  ; いったん作業領域に保存
025     BTFSC    PORTB,7             ;
026     GOTO     GetTable1
                 ; キーが押されてなかったらテーブルの前半を使用
027
028     MOVLW    8    ; キーが押されていたらテーブルの後半を使用
029     ADDWF    Wk,W                ;
030
031 GetTable1:
032     ADDWF    PCL,F      ; Wレジスタと PCを加算してジャンプ
033
034     RETLW    0x01
          ; PortBの最下位ビットから順に点灯するビット・パターン
035     RETLW    0x02
036     RETLW    0x04
037     RETLW    0x08
038     RETLW    0x10
039     RETLW    0x20
040     RETLW    0x40
041     RETLW    0x80
042
043     RETLW    0x80
          ; PortBの最上位ビットから順に点灯するビット・パターン
044     RETLW    0x40
045     RETLW    0x20
046     RETLW    0x10
047     RETLW    0x08
048     RETLW    0x04
049     RETLW    0x02
050     RETLW    0x01
051
052 ;.................................................
053 ; ソフト・タイマ
054 LongDelay:
055 ;.................................................
056
057     MOVF     PORTC,W
                  ; ポートCのデータを読み取ってカウンタ値とする
058     MOVWF    CountOuter         ; いったんCountOuterに格納
059
060 LoopOuter:
061     MOVLW    255                ; 値255をCountInnerに格納
062     MOVWF    CountInner
063
064 LoopInner:
065     NOP
066     NOP
067     DECFSZ   CountInner,F
068     GOTO     LoopInner
                 ; CountInnerがゼロになるまでループする
069
070     DECFSZ   CountOuter,F
071     GOTO     LoopOuter
                 ; CountOuterがゼロになるまでループする
072
073     RETLW 0
074
075 ;.................................................
076 ; メイン・プログラム
077 MainProc:
078 ;.................................................
079     CLRF     LEDPos     ; LEDの点灯位置メモリをクリアする
080     CLRF     PORTA      ; LEDドライブ・ポートを全部OFFにする
081
082 MainLoop:
083     CALL     LongDelay    ; ソフト・タイマ
084     INC      LEDPos       ; LEDの点灯位置を表すメモリを+1
085     MOVF     LEDPos,W     ; その値をWレジスタに格納
086     CALL     GetTable     ; LEDをドライブするデータを得る
087     MOVWF    PORTA        ; LEDドライブ・データを出力する
088
089     GOTO     MainLoop     ; ループする
090
091     .END
```

● 組み込みプロジェクトの作成

ソフトウェアの場合も，プロジェクトが開発の単位になります．最終的にFPGAの中に配置したCPUに関連づけられます．

DXPのメニューから「ファイル（F）」→「新規（N）」→「Project」→「Embedded Project」を選択し，組み込みプロジェクトの作成を行います．**図19**のように，プロジェクト名はデフォルトでEmbedded_Project1.PrjEmbになっています．メニューから「ファイル（F）」→「プロジェクトの保存」を選択し，「CQFpgaPicExample.PrjEmb」と名まえを変えて保存しておきます．

次に，ソース・ファイルを新しく作成します．コーディングを行うには，DXP 2004に付属しているエディタを使用します．CQFpgaPicExample.PrjEmbの文字の部分を右クリックして，「新規ドキュメントをプロジェクトに追加（N）」→「Assembly File」を選択すると，Source1.ASMというファイルが追加され，空のファイルが画面の右に表示されます．このファイルも名まえを「PicExample.ASM」に変更して保存しておきます（**図20**）．

● アセンブラやリンカの設定

次に，アセンブラやリンカのさまざまなオプションを設定します．

CQFpgaPicExample.PrjEmbの文字の部分を右クリックし，「プロジェクト・オプション（O）」を選択すると，**図21**のような画面が表示されます．

今回は使用するプロセッサとして，「TASKING TSK 165x」を指定します．ほかはデフォルトのままでも問題な

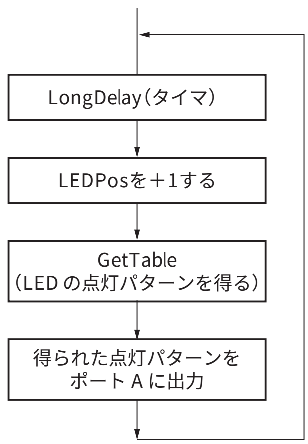


図18
メイン・プログラムのフローチャート
ソフトウェア・タイマとLEDの点灯パターンを作成するサブルーチンがある．



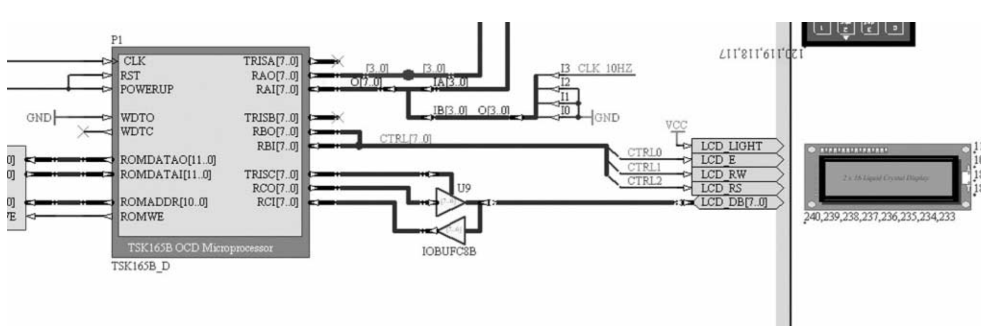
図19
アセンブラ・プロジェクトの作成
ソフトウェアの場合も，プロジェクトが開発の単位になる．

コラム　CPUコアと実際のマイコンの違い

PICマイコンと命令セット互換のCPUコア「TSK165A_D」は，実際のPICマイコンと異なり，入力ポート，出力ポート，入出力の切り替え端子が分かれています．

TSK165A_Dがポートの入出力を切り替えるTRIS命令を実行すると，TRISA/TRISB/TRISC端子からWレジスタの内容が出力されます．ポートを双方向で使用する場合には，これらの信号によって制御される入出力切り替え回路を用意します．

ポートを双方向で使用する例として，入出力共用の制御バスを持つLCDモジュールがあります．回路の例を**図E-1**に示します



図E-1　LCDモジュールとのインターフェース
RCポートを入出力ポートとして使用する．

く動作します．

● FPGAプロジェクトとの関連づけ

作成した組み込みプロジェクトが，どのFPGAプロジェクトのどのプロセッサに対応するのかを指定します．

FPGAプロジェクトは，あらかじめビルド(FPGAの回路データを完成)しておきます．プロジェクト画面の表示を「構成エディタ」に切り替えます．そして，組み込みプロジェクトであるCQFpgaPicExample.PrjEmbをCPUコア(U12［TSK165A_D］)の上にドラッグ・アンド・ドロップします(**図22**)．

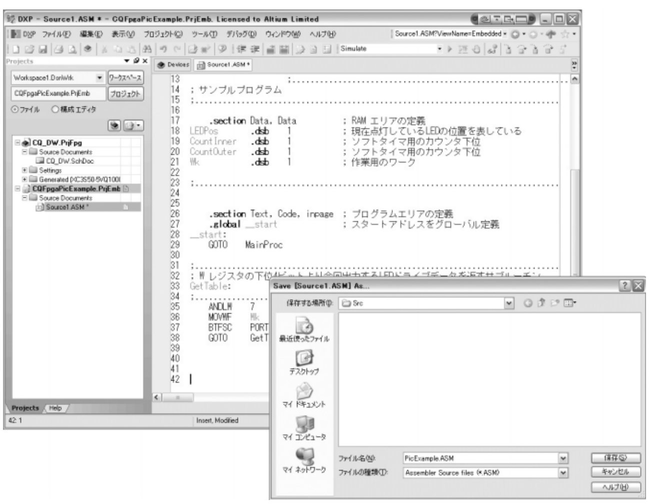


図20 ソフトウェアのコーディング
コーディングにはDXPに付属するエディタを使用する．ファイル名は「PicExample.ASM」とする．

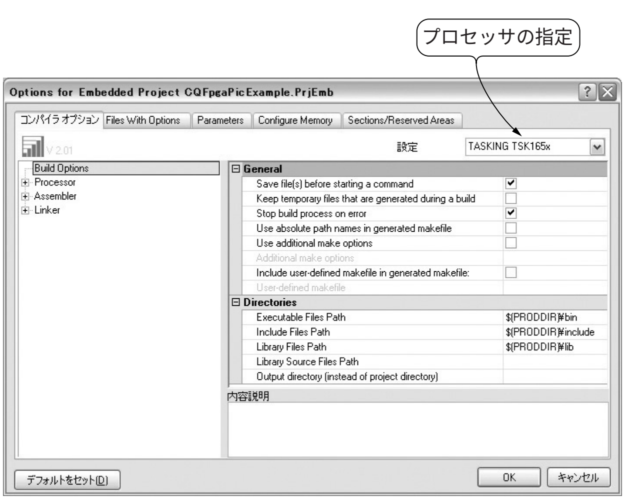


図21 アセンブラやリンカの設定
使用するマイコンの種類を指定する．

● アセンブルとリンク

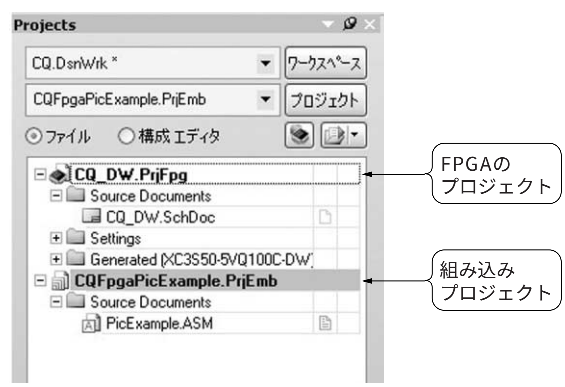
メニューから「プロジェクト(C)」→「Compile」→「Compile Embedded Project CQFpgaPicExample.PrjEmb」(または「Recompile Embedded Project Project CQFpgaPicExample.PrjEmb」)を選択して，アセンブルとリンクを行います．

FPGAプロジェクトと同様に，エラーがあればメッセージが表示されます．エラー・メッセージをクリックするとソース・ファイルのエラー箇所が表示されるので，誤りを修正します．

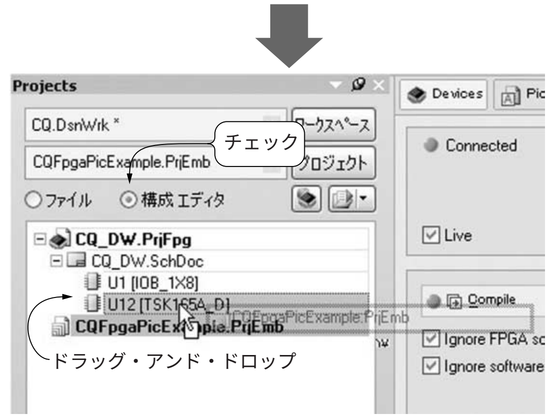
● シミュレーション・デバッグ

DXP 2004に付属するCPUコアの開発ツールには，シミュレーション・デバッガが付属しています．

メニューから「デバッグ(D)」→「現在のデバッグ」で表示される画面で「simulate」を選択し，［実行(R)］ボタンを押



(**a**) FPGAプロジェクトと組み込みプロジェクト



(**b**) 関連づけ

図22 FPGAプロジェクトとの関連づけ
組み込みプロジェクトであるCQFpgaPicExample.PrjEmbをCPUコア(U12［TSK165A_D］)の上にドラッグ・アンド・ドロップする．

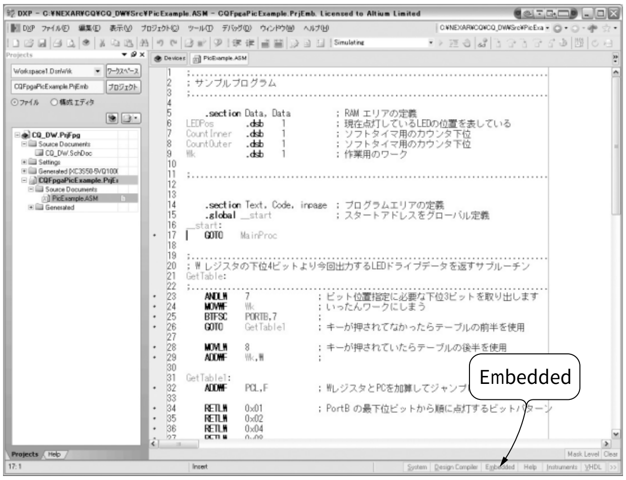


図23　シミュレーション・デバッガ
エディタの画面がそのままデバッガに移行する．

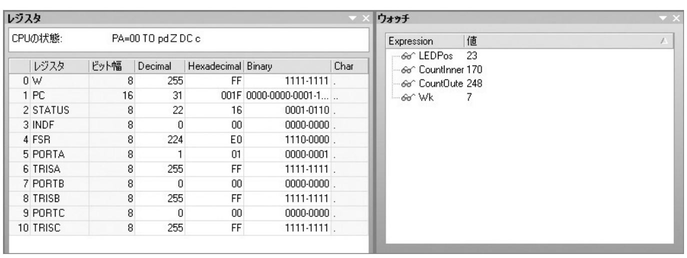


図24　レジスタ・ウィンドウとウォッチ・ウィンドウ
今回のソフトウェアのデバッグでは，レジスタ・ウィンドウとウォッチ・ウィンドウを並べて表示させるとよい．

すとシミュレーション・デバッガが起動します．

図23の画面の右下にある「Embedded」をクリックすると，メモリ・レジスタ，コール・スタック，ブレーク・ポイントなどを表示させるウィンドウを選択できます．必要なものだけを選択して表示し，デバックを行います．今回のソフトウェアの場合，**図24**のようにレジスタ・ウィンドウとウォッチ・ウィンドウを表示させるとよいでしょう．

ソフトウェアのデバッグでは，ソフトウェアの実行を一時停止するブレーク・ポイントをよく使います．このデバッガでは，ソース・プログラムの行番号の左側をクリックすることで，ブレークの設定/解除を行えます．今回のソフトウェアでは，二つのサブルーチンの出入り口の部分，およびLEDポート（PORTB）にデータを出力する部分でブレークし，Wレジスタが設計どおりのデータを保持していることを確認します（**図25**）．

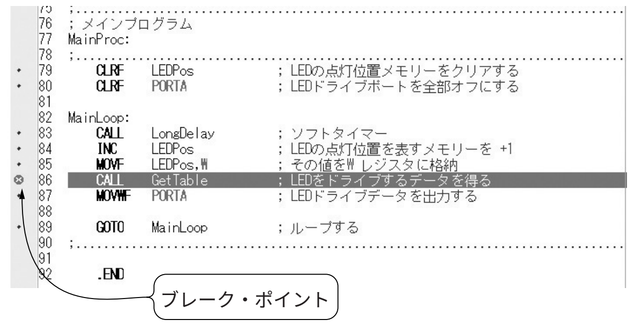


図25　ブレーク・ポイント
ソース・プログラムの行番号の左側をクリックすることで，ブレークの設定/解除を行える．

● 実機によるデバッグの前準備

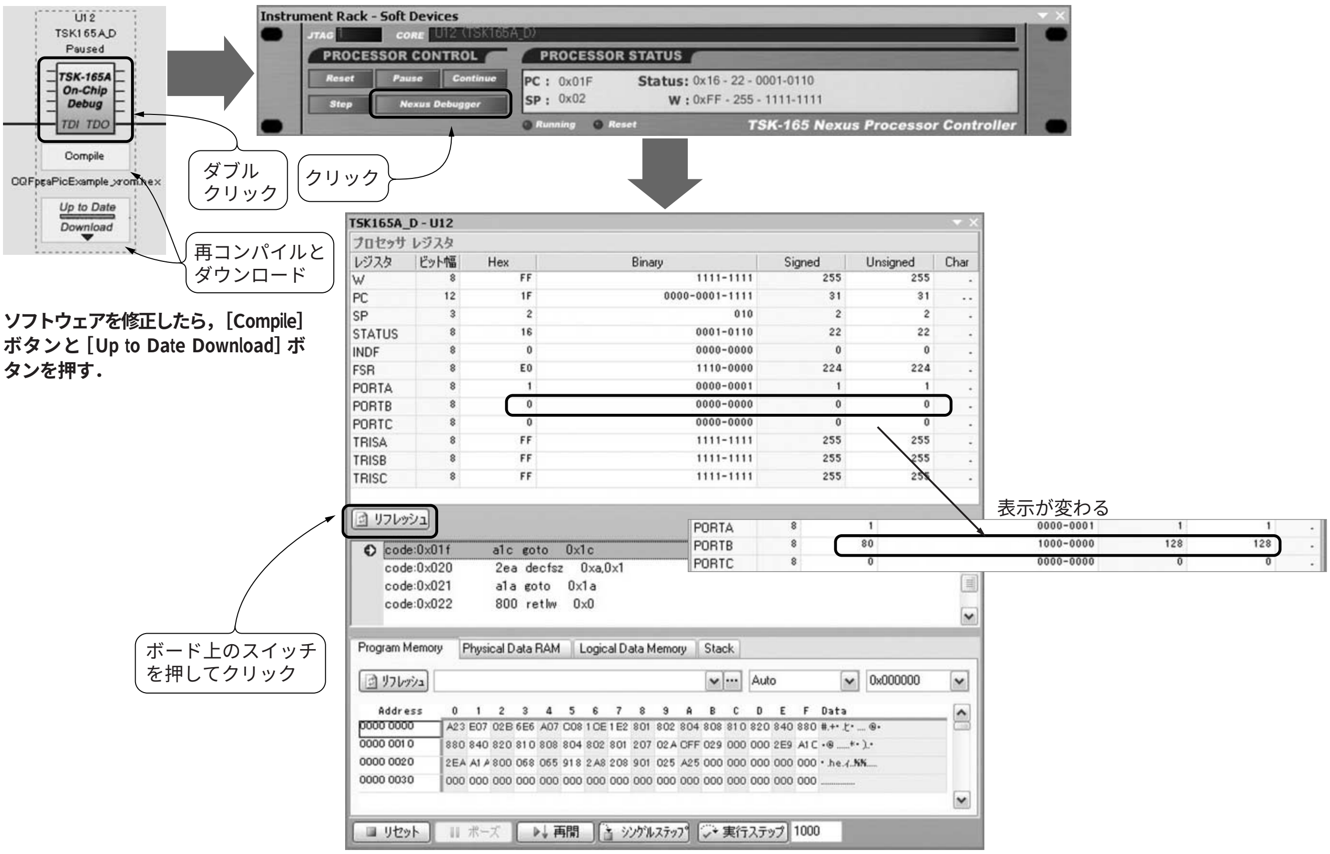
実機でソフトウェアを動作させる前に，ハードウェアが正常であることを確認しておきます．手順を以下に示します．

1）FPGAプロジェクトをコンパイルし，回路データをFPGAに書き込む．この段階で赤LEDが点灯，緑LEDが消灯になっていない場合（例えば二つとも薄暗く光った状態），FPGAに回路データが書き込まれていないか，または制約ファイルで指定したピン番号がまちがっていることが考えられる．
2）基板上のスイッチを押して赤LEDが消灯，緑点灯になることを確認する．
3）ソフトJTAGチェーンに表示されたTSK165A_Dのチップの絵をダブルクリックし，仮想計測器の一つである「TSK-165 Nexus Processor Controller」のパネルを表示する．［Nexus Debugger］ボタンをクリックすると，デバッガ画面が表示される．FPGAボード上のスイッチを押して［リフレッシュ］ボタンをクリックする．正しく動作している場合，PORTBのMSBが'1'に変化する（**図26**）．
4）仮想測定器であるDigitalIOに設定した値が，PORTCの入力データとして読み出せるかどうかを確認する．
5）LEDの制御ポートであるPORTAのデータ部分に任意のデータを書き込み，そのとおりに点灯するかどうかを確認する（**図27**）．

● 実機によるデバッグ

FPGAが動作した後は，実機によるデバッグを行うことができます．

メニューから「デバッグ」→「現在のデバックモード」→



ソフトウェアを修正したら，[Compile] ボタンと [Up to Date Download] ボタンを押す．

図26　TSK-165 Nexus Processor Controller

ソフトJTAGチェーンに表示されたTSK165A_Dのチップの絵をダブルクリックすると，「TSK-165 Nexus Processor Controller」のパネルが表示される．[Nexus Debugger] ボタンをクリックすると，デバッガ画面が表示される．FPGAボード上のスイッチを押して [リフレッシュ] ボタンをクリックする．正しく動作している場合，PORTBのMSBが‘1’に変化する．

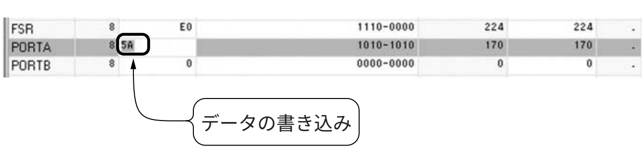


図27　LEDの制御ポートのデバッグ

PORTAのデータ部分に任意のデータを書き込む．

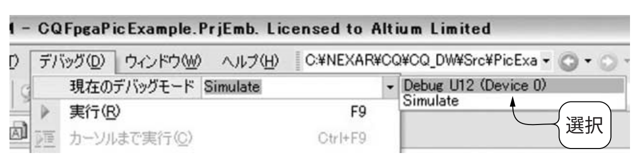


図28　シミュレーションと実機の選択

実機でデバッグを行うときには，「Debug U12 (Device 0)」(回路によって表示は変わる) を選択する．

「Debug U12 (Device 0)」を選択し，「デバッグ(D)」→「実行(R)」をクリックします(**図28**)．後の操作は，シミュレーション・デバッグの場合と同じです．

シミュレーション・デバッグを十分に行っていれば，実機では一発で動作するはずです．もし思いどおりに動作しない場合は，今までのチェックをもう一度繰り返してください．

● ソフトウェアの修正

デバッグや仕様変更の結果，ソフトウェアを修正したとき，アセンブルやリンクをやり直す必要があります．このとき，毎回組み込みプロジェクトを開く必要はありません．**図26**の[Compile] ボタンと [Up To Date Download] ボタンをクリックするだけで，FPGA内にあるCPUコアの命令メモリにソフトウェア・オブジェクト・コードがダウンロードされます．

● ほかのCPUコアを使う

今回はターゲットFPGAが小さいこともあり，PICという小型のCPUコアを使用しましたが，ほかのCPUコアであっても基本的な操作は同じです．あらかじめ用意されて

いるCPUコアとして，ほかに，

- TSK51x/TSK52x：8051相当のコア
- TSK80x：Z80相当のコア
- PowerPC：Power PC相当のコア
- TSK3000：MIPS3000相当のコア

があります．C言語もサポートされているので，より大規模なプログラムも開発可能です．今後，FPGAに搭載できるCPUコアの種類が増えれば，組み込みシステムの開発がより柔軟になることでしょう．

3. FPGAを搭載した基板の作成

剣持裕治

ここまではFPGA内部の開発に焦点を当てて説明してきましたが，FPGAを動作させるにはプリント基板が必要です．LSI設計者であってもある程度のプリント基板設計に関する知識は必要です．ここではFPGAの開発とデバッグが終了したものとして，このFPGAを搭載するプリント基板を製作する工程を体験してみましょう．

プリント基板の設計を詳細に解説しようとすると，小規模なボードであっても膨大な誌面を必要としてしまいます．そこで，ここではプリント基板設計の流れを理解できる範囲の解説にとどめます．

● PCBプロジェクトを作成する

設計が完了したFPGAといえども，単独では動作しません．FPGAを使用した製品を世に送り出すためには，FPGAを実際に実装し，電源を供給したり周辺I/Oと接続するためのプリント基板が必要です．

DXPで開発の終わったFPGAは，プロジェクトのデータのままプリント基板設計へ移行することができます．

まず，FPGAプロジェクトを開き，最上位階層の回路図を開きます．次にメニューから「ツール(T)」→「FPGA To PCBプロジェクト ウィザード(Z)」を選択し，**図29**のウィザードを実行します．FPGAの型名やPCBプロジェクト名などの質問に答えていけば，回路図が自動的に作成されます．各I/Oピンのネット名や入出力の方向，電源・GNDピン，特殊機能ピンなどがすべてそろったFPGAのシンボルが配置されます．

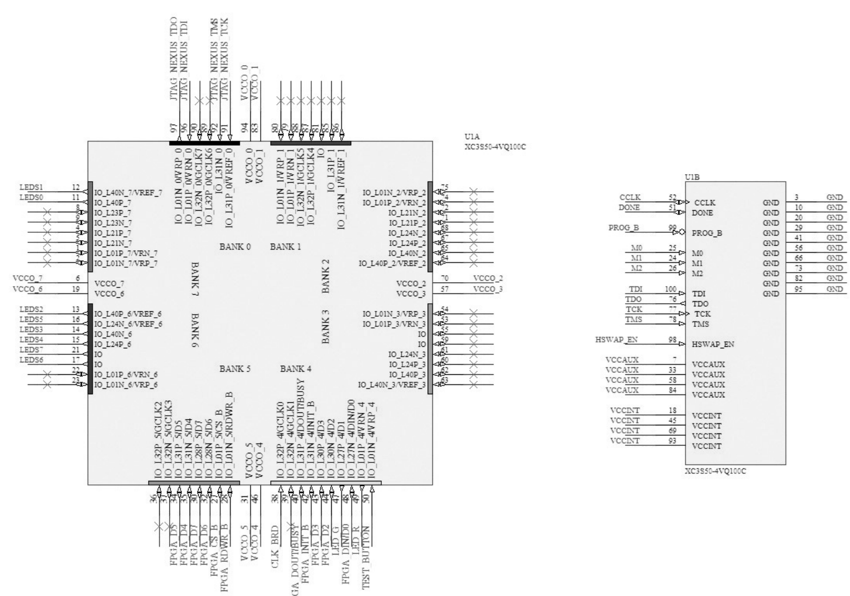
実際の回路図は，一つ上の階層で作成します(**図30**)．

● プリント基板上に部品を配置する

作成が終わったPCBプロジェクトの回路図からプリント基板のデータを作成します．細かい設計ルールや基板の属性などを設定するためのウィザードも用意されていますが，



図29
FPGA to PCBプロジェクト・ウィザード
FPGAの型名やPCBプロジェクト名などの質問に答えていけば，回路図が自動的に作成される．



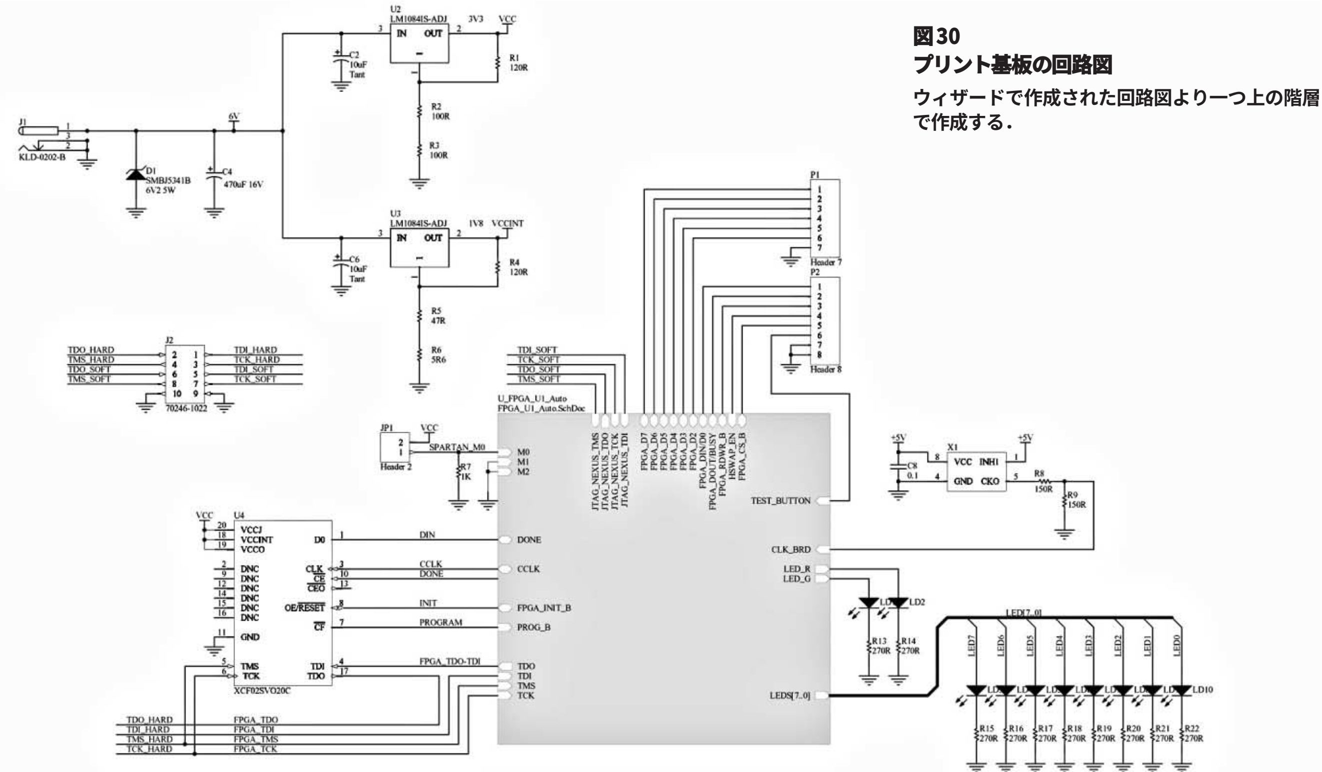


図30
プリント基板の回路図
ウィザードで作成された回路図より一つ上の階層で作成する．

今回は簡単に「ファイル（F）」→「新規（N）」→「PCB」を選択して，空のプリント基板を準備します．次に，「プロジェクト（C）」→「相違点の表示（S）」を選択して「詳細モード」にしてから，回路図とプリント基板の二つのドキュメントを比較します．比較して検出された差分は，ダイアログで表示されます．今回は空のプリント基板とでき上がった回路図との比較なので，すべてのネットと部品がプリント基板側に追加される変更が生じているはずです．変更はECO（設計変更指示）を介して更新され，プリント基板のデータにネット付きの部品が仮配置されています．プリント基板の外形を定義し，その外形内に部品を配置していきます．**図31**は部品配置が完了したところです．

部品の接続関係を考慮すると，これで完ぺきではありません．例えば，**図31**の上部のP1やP2，LEDからの配線に注目すると，かなり遠回りになり，むだの多い配線になることが予想されます．

FPGAの場合，I/Oピンにある程度の自由度があり，入れ替えが可能です．このようなむだが出るのは，FPGAの制約ファイルでピン配置を指定した際に，プリント基板に部品を配置した状態まで見通せなかったためです．

ここでFPGAの柔軟性を利用します．プリント基板上の

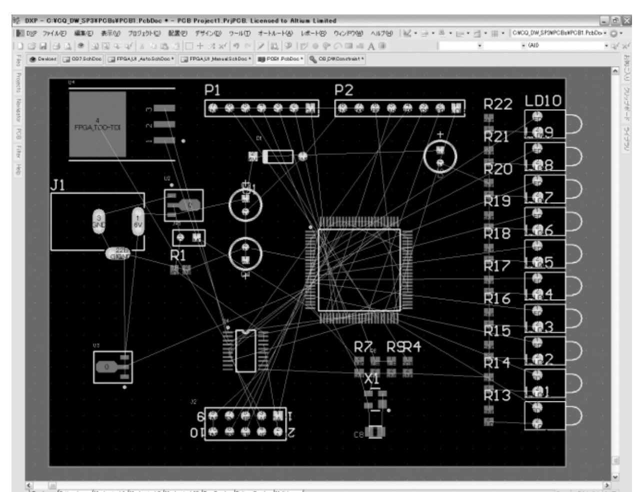


図31　部品の配置が完了したプリント基板
LEDやコネクタの位置は固定されている．これらに接続される配線の中には，遠回りしそうなものが見受けられる．

部品配置に配慮してFPGAのピン配置を変更し，FPGAの制約ファイルにその変更を反映させます．

まず，「ツール（T）」→「FPGAピンスワップ（W）」→「FPGAピンスワップ マネージャ（M）」を選択し，**図32**の画面を開きます．すべての「スワップグループ」を「1」とする設定を行います．これは，I/Oバンクなどを無視して，

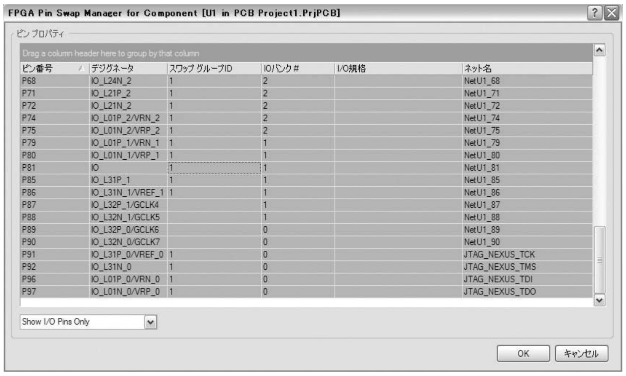


図32　ピンスワップ・マネージャ
ピンの入れ替えの設定を行う．

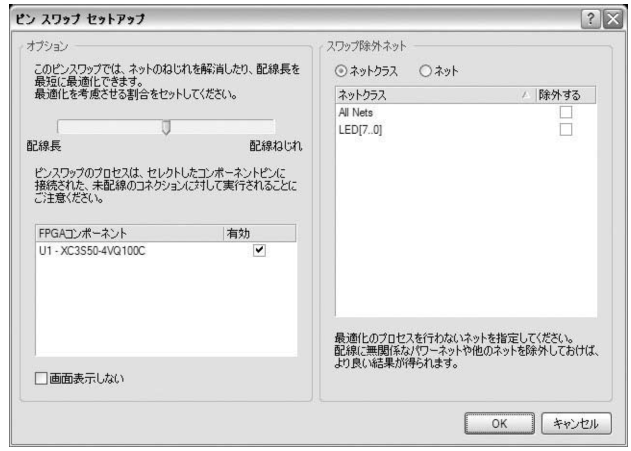


図33　ピン・スワップ・セットアップ
ピンの入れ替えにあたっての最適化オプションを設定する．

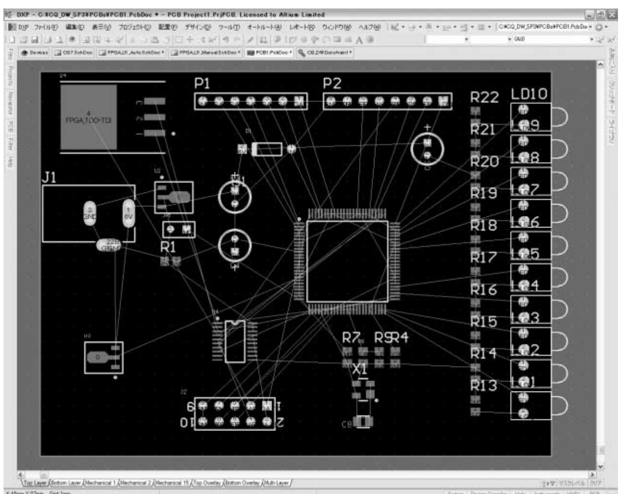


図34　配線の最適化
図31と比べると，配線にむだがなさそう．

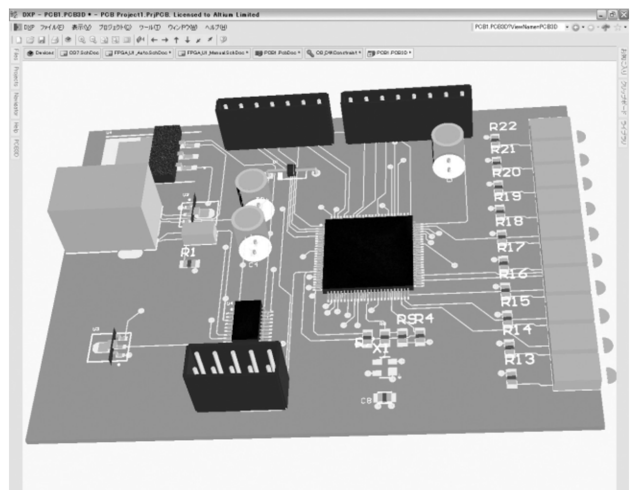


図35　プリント基板を3D表示
プリント基板に部品を実装したようすを3次元(3D)表示できる．

すべてのI/Oピンを入れ替え(スワップ)対象とする設定です．［OK］ボタンを押し，ダイアログを閉じ，「ツール(T)」→「FPGAピンスワップ(W)」→「オート(A)」を選択すると，**図33**の画面が開きます．さまざまな最適化オプションがありますが，今回はデフォルトで［OK］ボタンを押します．すると最適化がすぐに実行されます．そして**図34**に示すように，入れ替え可能なI/Oピンへの配線について，短くかつビアを必要としない整理された配線に最適化されました．

プリント基板に部品を配置した後で各I/Oピンからの接続を最適化できるということは，大きな利点です．というのは，余分なビアやむやみに長い配線あると，基板コストが上昇したり，設計品質が低下することがあるからです．

図35は自動配線を実施し，今回のプロジェクトのプリント基板を3次元(3D)表示させたところです．

けんもち・ゆうじ
もり・あきら

<著者プロフィール>

剣持裕治．猫アレルギーのぜんそく持ちにもかかわらず，ペルシャ猫9匹と同居する命知らずで知られる．Altium社のツールとはWindows初期のころからの長い付き合いになる．今回の執筆では，キーボード上に飛び乗る猫とベータ版の動作との格闘であったらしい．

森あきら．マイコンとは8080時代からの付き合い．今では同じ8ビットでも1けた以上速く，メモリも潤沢にある．だから，いい加減にプログラムを組んでも「動いてしまう」ため，くふうのないプログラムがはんらんしているような気がしている．